*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Feature Document

User Story #**194**

**Team Member:**

Gabriel Fernandez

**Product Owner(s)**:

Mohsen Taheri

**Mentor(s)**:

Mohsen Taheri

**Instructor**: Masoud Sadjadi

**User Story** Create infrastructure for e-mail validation

* As a developer, I want to have an infrastructure on the backend to be able to validate a user’s email, so that user emails are verified to be correct.

**Acceptance Criteria**

* Users’ emails are verified in the database
* There exists endpoints that the front end can call
* The backend is able to send emails to the user
* A new code is generated each time a validation email is sent

**Use Case** #**194 – Validate a user’s email**

**Actors**

Front-end developer

App user

**Entry Conditions**

Front-end developer has access to a terminal or an api to send http requests

Front-end developer has read the documentation and understands which api endpoints to hit and what data to send

App user has just registered with a new e-mail

**Flow of Events**

1. Use case begins when front-end developer sends an http request to the backend’s email endpoint with a user token as a header
2. The backend authenticates the user
3. The backend creates a randomly generated code and writes it on the user’s profile
4. The backend queries the database for the authenticated user’s email and sends an email to the app user with the code
5. The backend sends a response to the front-end developer saying that the email has been sent
6. The app user receives the code in his email and types it into the front end application
7. Front-end developer sends an http request to the backend’s email endpoint with a user token as a header and the user code as data
8. The backend authenticates the user
9. The backend checks the given code against the user profile’s code in the database
10. The codes match and the user profile is written as verified in the database; The front-end developer receives a response saying that the user has been validated, and the use case ends.

**Alternate Flow of Events**

* 2a.
  + The backend determines the token does not belong to an existing user and the use case ends.
* 8a.
  + The backend determines the token does not belong to an existing user and the use case ends.
* 10a.
  + The codes do not match; The front-end developer receives a response saying that the user could not be validated, and the use case ends.

**Use Case Diagram**



**Sequence Diagram**



**Class Diagram**



**Unit Test**

**Test Case 1 (Sunny Day)**

**Purpose**

* Ensure backend is able to create codes and send emails to users

**Precondition**

* Http request is made to the server, using the send\_email URL

**Input**

* User token

**Expected Result**

* Backend sends an email with a code to the user’s email address

**Actual Result**

* Email with code received

**Test Case 2 (Sunny Day)**

**Purpose**

* Ensure backend verified a user given a correct code

**Precondition**

* Http request is made to the server, using the verify URL & user’s code

**Input**

* User token

**Expected Result**

* User verified

**Actual Result**

* User verified

**Test Case 3 (Rainy Day)**

**Purpose**

* Ensure backend does not verify incorrect codes

**Precondition**

* Http request is made to the server, using the verify URL & user’s incorrect code

**Input**

* User token

**Expected Result**

* User could not be verified

**Actual Result**

* User could not be verified

**Visual User Guide**





